```
// This source code is subject to the terms of the Mozilla Public License 2.0 at https://mozilla.org/M
// © pikusov
//
// Diagonal Supports and Resistances
//
// Classic diagonal support and resistance based on pivot points.
// As a result, they form triangles, wedges, channels and other patterns.
// Realtime update up to 1 second chart.
//
// Input parameters:
// History Bars Back: Number of bars back to find high and low points
// Please note that History Bars Back more than 300 may cause errors
// Resolution: The power of high and low points to take into drawing
//
// You can enable alerts for crossing supports and resistances in the Alerts menu
// https://www.tradingview.com/support/solutions/43000595315-how-to-set-up-alerts
//
//@version=4

var int max_bars_back = 5000
study("Support Resistance Diagonal", overlay=true, max_bars_back = max_bars_back)


var int history_bars = input(title="History bars back", type=input.integer, defval=300)
col_sup = color.new(#17ff27, 50)
style_sup = line.style_solid
col_res = color.new(#ff77ad, 50)
style_res = line.style_solid

// Функция вычисляет цену в точке t3 для линии,
// заданной первыми четырьмя координатами (t1, p1, t2, p2)
price_at(t1, p1, t2, p2, t3) =>
    p1+(p2-p1)*(t3-t1)/(t2-t1)

// Alerts
if(1 == 1)
    alert('test')
// округление
round_to_tick(x)=>
    mult = 1 / syminfo.mintick
    value = ceil(x*mult)/mult

// Тут храним линии для удаления при появлении нового бара
var line[] supports = array.new_line()
var line[] resistances = array.new_line()
var label[] labels = array.new_label()

fire_alert_sup = false
fire_alert_res = false
fire_alert_sup := false
fire_alert_res := false
// Удаляем прошлые линии и заодно вызываем алерты

```

```
line temp_line = na
if array.size(supports)>0
    for i = array.size(supports)-1 to 0
        temp_line := array.get(supports, i)
        if (low[1]>line.get_price(temp_line, bar_index-1)) and (close<line.get_price(temp_line, bar_ir
            fire_alert_sup := true
        line.delete(temp_line)
        array.remove(supports, i)
if array.size(resistances)>0
    for i = array.size(resistances)-1 to 0
        temp_line := array.get(resistances, i)
        if (high[1]<line.get_price(temp_line, bar_index-1)) and (close>line.get_price(temp_line, bar_i
            fire_alert_res := true
        line.delete(temp_line)
        array.remove(resistances, i)
label temp_label = na
if array.size(labels)>0
    for i = array.size(labels)-1 to 0
        temp_label := array.get(labels, i)
        label.delete(temp_label)
        array.remove(labels, i)


alertcondition(fire_alert_sup, "Diagonal Support Alert",  "Diagonal support crossed down")
alertcondition(fire_alert_res, "Diagonal Resistance Alert",  "Diagonal resistance crossed up")

// Определяем экстремумы
min_values = low
max_values = high
x1 = input(title="Resolution (bars)", type=input.integer, defval=6)
x2 = round(x1/2)
int minimums = 0
minimums := lowestbars(min_values, x1) == -x2 ?x2:minimums[1]+1

int maximums = 0
maximums := highestbars(max_values, x1) == -x2 ?x2:maximums[1]+1


int minimum1 = 0
int minimum2 = 0
int maximum1 = 0
int maximum2 = 0
int medium = 0
// Поддержка
if  barstate.islast
    //label.new(bar_index, close , style=label.style_labeldown, text=timeframe.period, color=color.new
    line last_line = na
    label last_label = na
    for k1 = 0 to 50
        if(minimum1>=history_bars)
            break
        minimum1 := minimum1 + minimums[minimum1]
        minimum2 := minimum1*2
        for k2 = 0 to 50
            if(minimum2>=minimum1*8 or minimum2>=history_bars)
```

```
    break
minimum2 := minimum2 + minimums[minimum2]

if(minimum1>=history_bars or minimum2>=history_bars)
    break

bar1 = bar_index-minimum1
bar2 = bar_index-minimum2

price1 = low[minimum1]
price2 = low[minimum2]

current_price = price_at(bar2, price2, bar1, price1, bar_index)
// Если поддержка проходит ниже текущей цены
if(current_price < high[1])

    // проверяем пересечения
    crossed = 0
    medium := 0
    for k3 = 0 to 50
        if(medium >= minimum2)
            break
        medium := medium + minimums[medium]
        if(medium >= minimum2)
            break
        if price_at(bar2, price2, bar1, price1, bar_index-medium)>min(open[medium], close[
            crossed := 1
            break

    // если нет пересечений
    if crossed == 0 // and overtilt == 0
        // сравниваем с прошлой созданной линией
        if(not na(last_line))
            last_price = price_at(line.get_x1(last_line), line.get_y1(last_line), line.get
            if(bar1 == line.get_x2(last_line))
                if(current_price > last_price)
                    line.set_xy1(last_line, bar2, price2)
                    line.set_xy2(last_line, bar1, price1)
                    line.set_color(last_line, col_sup)
                    label.set_xy(last_label, bar_index, current_price)
                    label.set_text(last_label, tostring(round_to_tick(current_price)))
                    true
            else
                last_line := line.new(bar2, price2, bar1, price1, extend=extend.right, col
                last_label := label.new(bar_index, current_price, color=col_sup, style=lab
                array.push(labels, last_label)
                array.push(supports, last_line)
                true
        // добавляем линию
        else
            last_line := line.new(bar2, price2, bar1, price1, extend=extend.right, color=c
            last_label := label.new(bar_index, current_price, color=col_sup, style=label.s
            array.push(labels, last_label)
            array.push(supports, last_line)
            true
```

```
last_line := na
last_label := na
for k1 = 0 to 100
    if(maximum1>=history_bars)
        break
    maximum1 := maximum1 + maximums[maximum1]
    maximum2 := maximum1*2
    for k2 = 0 to 50
        if(maximum2>=maximum1*8 or maximum2>=history_bars)
            break
        maximum2 := maximum2 + maximums[maximum2]

        if(maximum1>=history_bars or maximum2>=history_bars)
            break

        bar1 = bar_index-maximum1
        bar2 = bar_index-maximum2

        price1 = high[maximum1]
        price2 = high[maximum2]

        current_price = price_at(bar2, price2, bar1, price1, bar_index)
        // Если сопротивоение проходит выше текущей цены
        if(current_price > low[1])

            // проверяем пересечения
            crossed = 0
            medium := 0
            for k3 = 0 to 100
                if(medium >= maximum2)
                    break
                medium := medium + maximums[medium]
                if(medium >= maximum2)
                    break
                if price_at(bar2, price2, bar1, price1, bar_index-medium)<max(open[medium], close[
                    crossed := 1
                    break

            // если нет пересечений
            if crossed == 0 // and overtilt == 0
                // сравниваем с прошлой созданной линией
                if(not na(last_line))
                    last_price = price_at(line.get_x1(last_line), line.get_y1(last_line), line.get
                    if(bar1 == line.get_x2(last_line))
                        if(current_price < last_price)
                            line.set_xy1(last_line, bar2, price2)
                            line.set_xy2(last_line, bar1, price1)
                            line.set_color(last_line, col_res)
                            label.set_xy(last_label, bar_index, current_price)
                            label.set_text(last_label, tostring(round_to_tick(current_price)))

                            true
                        else
                            last_line := line.new(bar2, price2, bar1, price1, extend=extend.right, col
```

```
            last_label := label.new(bar_index, current_price, color=col_res, style=lab
            array.push(labels, last_label)
            array.push(resistances, last_line)
            true
    // добавляем линию
    else
        last_line := line.new(bar2, price2, bar1, price1, extend=extend.right, color=c
        last_label := label.new(bar_index, current_price, color=col_res, style=label.s

        array.push(labels, last_label)
        array.push(resistances, last_line)
        true
```
